

# Re: assignment in a for loop

---

*Source:* <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2006-05/msg02565.html>

---

- *From:* Ben Finney <[bignose+hates-spam@xxxxxxxxxxxxxxxxxx](mailto:bignose+hates-spam@xxxxxxxxxxxxxxxxxx)>
  - *Date:* Wed, 17 May 2006 15:05:29 +1000
- 

"MackS" <[mackstevenson@xxxxxxxxxxx](mailto:mackstevenson@xxxxxxxxxxx)> writes:

```
l = [1,2]
for i in l:

    ... i = i + 1
    ...

    1

[1, 2]
```

I understand (I think!) that this is due to the fact that in Python what looks like "assignment" really is binding a name to an object. The result is that inside the loop I am creating an object with value (i+1) and then "pointing" the name i at it. Therefore, the object to which i previously pointed (an element of list l) remains unchanged.

That's a fair explanation, yes.

Two brief questions:

- 1) Is what I wrote above (minimally) correct?

Correct for what? You can tell if it's *\*syntactically\** correct by simply running it.

As for any other "correct", define that. Does it do what you want it to do?

- 2) Independently of the answer to 1, is there a way for me to assign to elements of a list inside a loop and without resorting to C-style ugliness of

## Re: assignment in a for loop

```
for i in range(len(l))
    l[i] = l[i] + 1
```

You can build a new list from your operations on the old one.

```
new_list = []
for x in old_list:
    new_list.append(x+1)
```

You can also do it more succinctly with a list comprehension expression.

(Note: not using a list comprehension.)

What's preventing the use of list comprehensions?

```
new_list = [x+1 for x in old_list]
```

```
--
\ "Smoking cures weight problems. Eventually." -- Steven Wright |
\ |
_o_ |
Ben Finney
```

.